Realizing Unequal Error Correction for NAND Flash Memory at Minimal Read Latency Overhead

Jiangpeng Li, Kai Zhao, Jun Ma, and Tong Zhang

***Abstract*—In NAND Flash memory, all pages have the same storage capacity and hence accommodate the same amount of redundancy in support of error correction. In current practice, user data in all the pages are protected by the same error correction code. However, different types of pages in multibit per cell memory have largely different bit error rates, for which appropriate unequal error correction can achieve a better utilization of memory redundancy and hence improve program/erase (P/E) cycling endurance. Nevertheless, a straightforward realization of unequal error correction suffers from severe memory read latency penalty. This brief presents a design strategy to implement unequal error correction through concatenated coding, which can well match the unequal error rates among different types of pages at minimal memory read latency penalty. Based on measurement results from commercial sub-22-nm 2 bits/cell NAND Flash memory chips, we carried out simulations from both the coding and storage system perspectives, and the results show that this design strategy can improve the P/E cycling endurance by 20% and only incur less than 7% increase of storage system read response time at the end of Flash memory lifetime with the P/E cycling of around 1800.**

***Index Terms*—Error correction coding (ECC), NAND Flash memory.**

I. Introduction

NAND Flash memory data access is performed in the unit of page, and all the pages have the exactly same storage capacity and hence accommodate the same amount of redundancy in supporting error correction. In current practice, all the user data are protected by the same error correction code (ECC), and all the pages store the same number of equal-sized ECC codewords. Such equal error correction can simplify the Flash memory data management and ensure that we only need to read one page in order to recover the user data being protected by a single ECC codeword. For multibit per cell NAND Flash memory, different bits within each memory cell belong to different pages. This makes different types of pages have largely different bit error rates [1], leading to a mismatch with the equal error correction. As a result, available redundancy in NAND Flash memory is not fully utilized.

Manuscript received November 21, 2013; accepted March 12, 2014. Date of publication April 21, 2014; date of current version May 14, 2014. This work was supported in part by the Research Fund for the Doctoral Program of Higher Education of China under Grant 20110073110055 and in part by the National Science Foundation under Grants 1162152 and 1162165. This brief was recommended by Associate Editor Z. Wang.

J. Li and J. Ma are with Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: lijiangpeng@sjtu.edu.cn; majun@sjtu.edu.cn).

K. Zhao and T. Zhang are with Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: tzhang@ecse.rpi.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2014.2312640

To address this issue, we may employ unequal error correction among different types of pages, e.g., for multilevel per cell (MLC) NAND Flash memory, we may allocate more coding redundancy for protecting upper pages and meanwhile accordingly reduce the coding redundancy for protecting lower pages. Since all the pages have the same storage capacity, a certain portion of upper page coding redundancy has to be stored in lower pages. Hence, if upper pages are simply protected by a single ECC, we have to access one lower and one upper page when we need to read user data in an upper page. This will largely degrade memory read latency and, hence, system read response time. To address this read latency overhead issue, we propose a partial concatenated coding design strategy to protect user data stored in upper pages. When using this technique, reading user data being stored in upper pages does not always incur the lower page read, and the memory read latency overhead only gradually increases as program/erase (P/E) cycling gradually wears out Flash memory cells. Based on measurement results from sub-22-nm MLC Flash memory chips, we carried out simulations from both the coding and storage system perspectives, and the results show that this design strategy can improve the P/E cycling endurance by 20% at the cost of less than 7% increase of storage system read response time at the end of Flash memory lifetime with the P/E cycling of around 1800.

II. Background and Motivations

Memory cells on each NAND Flash memory die are organized in a plane⇒block⇒wordline hierarchy: each memory die contains few independent planes, each plane consists of a large number of blocks, each block contains a number of wordlines, and each wordline drives a very large number of memory cells. Since memory cells driven by the same wordline can be programmed or read simultaneously, NAND Flash memory handles data programming and read in the unit of page. For multibit per cell NAND Flash memory, different bits within each memory cell belong to different pages. This can be illustrated in Fig. 1 for MLC Flash memory, where the most significant bit and the least significant bit belong to a lower and an upper page. Such a multipage data organization can reduce memory read latency, e.g., as illustrated in Fig. 1, reading a lower (or upper) page only involves one (or two) sensing cycle with the reference voltage of V_{lower} (or V_{upper1} and V_{upper2}). In comparison, if both bits within one memory cell belong to the same page, we have to spend three sensing cycles to read one page.

As technology scales down, NAND Flash memory increasingly relies on ECC to ensure the data storage integrity. Therefore, Flash memory manufacturers have to fabricate enough number of memory cells along each wordline to accommodate both user data and coding redundancy. With the multipage

1549-7747 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

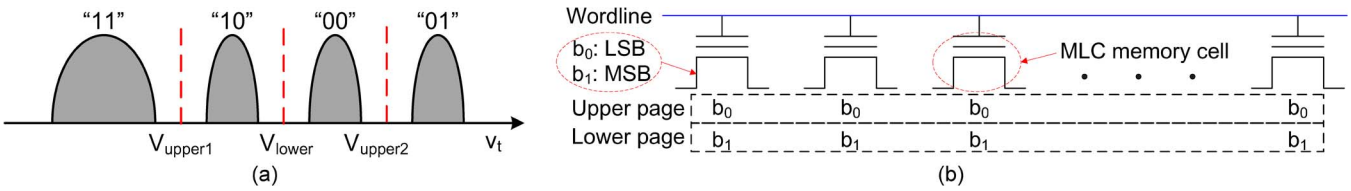

Fig. 1. Illustration of (a) data storage within each MLC memory cell and (b) multipage data organization for MLC memory cells on each wordline.

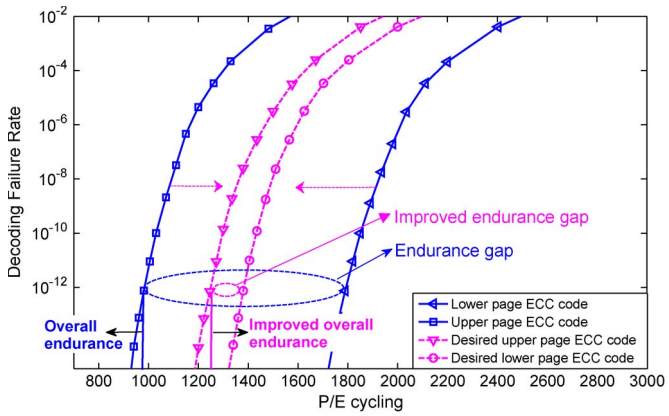

Fig. 2. Illustration of the endurance gap between upper and lower pages.

data organization, lower and upper pages have the exactly same size (as illustrated in Fig. 1) and accommodate the same amount of coding redundancy. Unfortunately, such an equal error protection does not well match to the unequal error rate among different pages, e.g., upper pages are subject to higher bit error rates than lower pages due to the nature of intracell bit mapping [1]. Therefore, with the same coding redundancy, upper pages cannot survive as many P/E cycling as lower pages, as illustrated in Fig. 2, and overall P/E cycling endurance is limited by that of upper pages. This clearly results in a suboptimal use of coding redundancy.

III. Proposed Design Solution

To reduce the redundancy usage suboptimality, we should allocate different amounts of coding redundancy to lower and upper pages so that the error correction strength can better match to their unequal bit error characteristics. This is referred to as unequal error correction, which clearly can achieve a higher P/E cycling endurance than conventional equal error correction. In MLC NAND Flash memory, assume that each wordline drives $N + R$ memory cells, aiming to accommodate $2N$-bit user data and $2R$-bit coding redundancy, and let $r_{\text{norm}} = N/(N + R)$. In current practice, all the pages are protected by the same rate-r_{norm} code. To realize unequal error correction, we should allocate more coding redundancy for protecting upper pages and meanwhile accordingly reduce the coding redundancy for protecting lower pages.

In spite of its simple concept, a straightforward realization of unequal error correction can largely degrade memory read latency. In the most straightforward manner, we use a stronger rate-r_u code to protect upper pages and a weaker rate-r_l code to protect lower pages, where $r_u < r_{\text{norm}} < r_l$ and $r_{\text{norm}} = (r_u + r_l)/2$. Since all the pages have the same size, a certain portion of the coding redundancy of rate-r_u codewords has to be stored in lower pages. As a result, we have to fetch both lower and upper pages whenever we need to read user data from

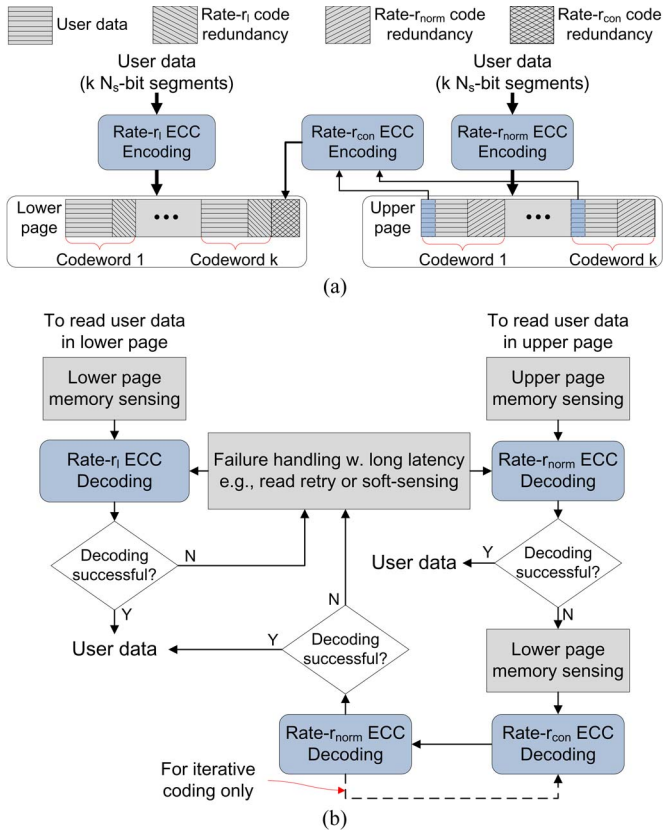

Fig. 3. Illustration of the proposed design solution. (a) Encoding and storage diagram. (b) Decoding flow diagram.

an upper page. Our measurements with sub-22-nm MLC Flash memory show that memory sensing latency is 41 and 55 μs for lower and upper pages, respectively. If we use the aforementioned straightforward realization, it will take a total sensing latency of 96 μs to read user data stored in an upper page.

To address this read latency issue, we propose a partially concatenated coding design strategy that employs concatenated coding to protect user data stored in upper pages. This can be illustrated in Fig. 3(a). Since ECC decoder implementation complexity tends to be proportional to the codeword length, current design practice always partitions the user data to be stored in each page into few equal-sized segments (e.g., 1 or 2 kB) in order to maintain a reasonable ECC decoder implementation cost. Let k and N_s denote the number of segments and the number of user data bits in each segment, respectively. For user data to be stored in a lower page, each N_s-bit segment is encoded with the rate-r_l code, and all the k rate-r_l codewords are stored in one lower page. Since $r_l > r_{\text{norm}}$, there are u_r-bit unoccupied space in the lower page, where

$$u_r = k \cdot N_s \cdot \frac{r_l - r_{\text{norm}}}{r_l \cdot r_{\text{norm}}}. \tag{1}$$

For user data to be stored in an upper page, each N_s-bit segment is first encoded with the rate-$r_{\rm norm}$ code, and all the k rate-$r_{\rm norm}$ codewords can exactly fit into one upper page. Meanwhile, we take N_s/k bits from each N_s-bit segment to form a new N_s-bit segment, encode this new segment with a rate-$r_{\rm con}$ code, and store its coding redundancy into the lower page. Since its coding redundancy should be equal to u_r defined in (1), we have

$$r_{\rm con} = \frac{r_l \cdot r_{\rm norm}}{r_l \cdot r_{\rm norm} + k \cdot (r_l - r_{\rm norm})}. \quad (2)$$

Compared with the aforementioned straightforward realization, this proposed design approach can very effectively reduce the read latency penalty. To read user data being stored in upper pages, we always first read and decode the rate-$r_{\rm norm}$ codewords from upper pages, and only if the rate-$r_{\rm norm}$ code decoding fails, we read the extra coding redundancy from lower pages. The overall flow diagram is shown in Fig. 3(b). To read user data stored in lower pages, we carry out normal hard-decision memory sensing and rate-r_l code decoding, and a decoding failure will invoke failure handling operation at a long memory sensing latency, e.g., Flash memory read-retry or fine-grained soft-decision memory sensing (if the ECC supports soft-decision decoding). To read user data stored in upper pages, we first carry out normal hard-decision memory sensing and rate-$r_{\rm norm}$ code decoding. In case of decoding failure, the concatenated decoding will be invoked, i.e., we will read the coding redundancy of the rate-$r_{\rm con}$ stored in the lower page and carry out rate-$r_{\rm con}$ code decoding and then rate-$r_{\rm norm}$ code decoding (there could be a few iterations). If it still fails, the failure handling operation will be invoked. Due to the use of stronger concatenated coding for the upper page and, meanwhile, weaker code for the lower page, the endurance gap between upper and lower pages will accordingly reduce. This can lead to the improved overall P/E cycling endurance, as illustrated in Fig. 2.

The previously presented design strategy has two advantages: 1) It can improve the Flash memory P/E cycling endurance at minimal memory sensing latency overhead. The memory sensing latency overhead naturally adapts to the memory wear-out. 2) All the codes (i.e., the rate-r_l, rate-$r_{\rm norm}$, and rate-$r_{\rm con}$ codes) have the same amount of user data, which can minimize the ECC decoder implementation cost [2]. For its practical realization, one key issue is choosing appropriate code rates r_l and $r_{\rm con}$ so that the data storage integrity of the user data stored in lower and upper pages is as close as possible. We note that r_l and $r_{\rm con}$ are dependent on each other, as shown in (2). This should be addressed by extensive simulations over a wide range of possible code rate values. As shown in Fig. 2, the optimal r_l and $r_{\rm con}$ should be able to minimize the endurance gap between the lower and upper pages.

IV. Experiment Results

A. Simulation Environments

To quantitatively evaluate the effectiveness of this design strategy, we carried out simulations from both the coding and storage system perspectives. Due to their superior error correction capability and recent success in hard disk drives, low-density parity-check (LDPC) codes have attracted much attention [3]–[7] and are seriously considered as the choice of ECC for solid-state drives (SSDs). Hence, we choose LDPC codes and primarily focus on quasi-cyclic LDPC codes in our evaluation. To quantify the relationship between the P/E cycling and the memory raw bit error rate, we carried out measurements commercial sub-22-nm MLC NAND Flash memory chips, where each page stores 9 kB, including 8-kB user data and 1-kB coding redundancy. Hence, the code rate $r_{\rm norm}$ is 8/9. A total of 2048 pages (hence 18 MB) randomly chosen from several chips are used to carry out raw bit error rate versus P/E cycling measurements.

We further evaluate the impact of extra memory sensing latency overhead on the storage system performance through trace-based simulations using the SSD module [8] in DiskSim [9]. We set that the SSD has eight channels, and each channel connects to eight Flash chips. Each Flash chip contains two dies that share an 8-bit IO bus and a number of common control signals, and each die contains four planes, each plane contains 2048 blocks, and each block contains 64 pages.

B. Coding Simulation Results

Based on the measured bit error rate under different P/E cycling, we carried out LDPC decoding simulations to determine the code rates r_l and $r_{\rm con}$. Recall that $r_{\rm norm}$ is 8/9. We consider two scenarios where each codeword protects 1- or 2-kB user data. All the decoding is done with a min-sum decoding algorithm with maximal five iterations and hard-decision memory sensing [2]. For the concatenated code decoding, the rate-$r_{\rm norm}$ code decoding and the rate-$r_{\rm con}$ code decoding can iterate in order to improve the decoding performance. Based on the simulation results, we observe that r_l of 10/11 appears to be the best choice. According to (2), given $r_l = 10/11$, $r_{\rm con}$ is 5/6 and 10/11 for 1- and 2-kB cases, respectively. Fig. 4 shows the representative simulation results. Fig. 4(a) and (b) shows the simulation results when concatenated coding uses $r_l = 10/11$ for 1- and 2-kB cases. As shown in the figures, conventional design practice (i.e., both lower and upper pages use the same rate-8/9 code) results in a significant gap between the achievable P/E cycling endurance of lower and upper pages.

As shown in Fig. 4(a) and (b), once we use rate-10/11 code for lower pages (i.e., set r_l as 10/11) and meanwhile use a concatenated coding with $r_{\rm con} = 5/6$ (1 kB case) and $r_{\rm con} = 10/11$ (2 kB case) for upper pages, lower and upper pages can achieve almost the same P/E cycling endurance (i.e., the endurance gap between the lower and upper pages is almost eliminated). At the same decoding failure rate, the improved P/E cycling numbers are about 20% for both of the 1- and 2-kB cases. In addition, the simulation results clearly show that a longer codeword length (i.e., 2 kB user data per codeword in this study) can noticeably improve the overall P/E cycling endurance than a shorter codeword length (i.e., 1 kB user data per codeword in this study). This is because ECC with a longer codeword length tends to have a stronger error correction strength. Nevertheless, a longer codeword length results in a higher decoder implementation cost. This directly leads to a performance versus implementation cost tradeoff, and we will present the decoder implementation cost results in Section IV-C.

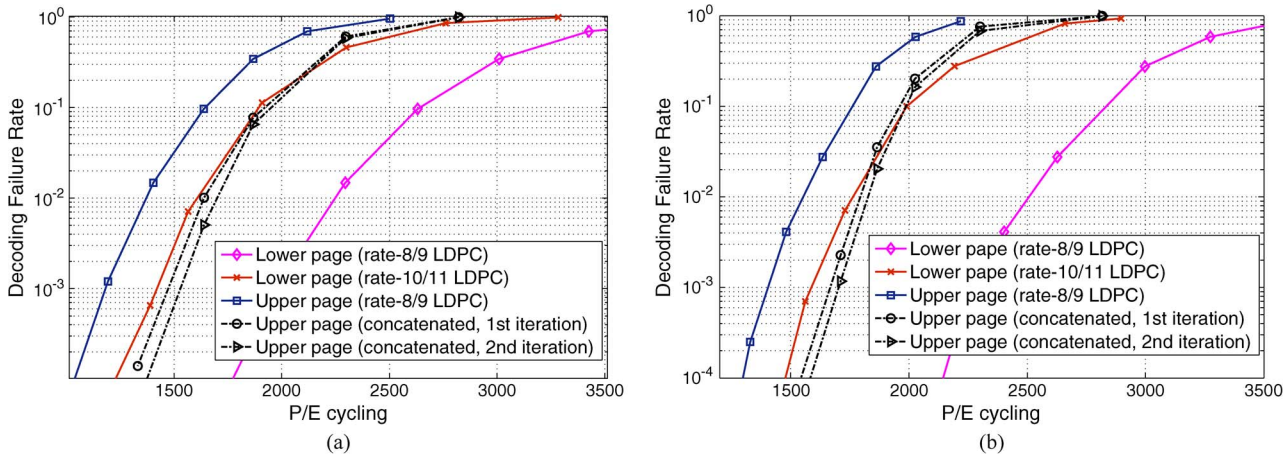

Fig. 4. Representative simulation results. (a) Concatenated coding with $r_l = 10/11$ (1 kB). (b) Concatenated coding with $r_l = 10/11$ (2 kB).

TABLE I
SILICON COST OF THE LDPC DECODERS AT 65-nm TECHNOLOGY

	1kB	2kB
Equivalent gate counts	270.7k	272.1k
Memory	26.35k bytes	48.95k bytes
Max. working frequency	800MHz	800MHz
Throughput	1GB/s	1GB/s
Power consumption	95 mw	98 mw

TABLE II
POWER CONSUMPTION (mW) ESTIMATION OF THE ECC DECODING

P/E cycling	500	1000	1400	1600	1800	2000
1kB (conventional)	40.2	64.3	110.6	-	-	-
1kB (proposed)	51.0	77.4	136.5	164.0	298.5	-
2kB (conventional)	31.5	56.3	105.5	123.3	-	-
2kB (proposed)	38.1	67.1	128.4	155.3	274.5	422.8

'-' means the scheme fails to work under the corresponding P/E cycle.

C. Silicon Cost Analysis

As pointed out earlier, this proposed design involves a tradeoff between the performance and LDPC code decoder implementation cost. Hence, we further carried out LDPC code decoder application specified integrated circuit design. We use the Synopsys synthesis tool set and 65-nm technology library. The decoder uses layered min-sum decoding algorithm with a partially parallel architecture [2], and we designed decoders for 1- and 2-kB cases. For a fair comparison between the 1- and 2-kB cases, both decoders have the same decoding computational parallelism in order to achieve the same 1-GB/s decoding throughput under five decoding iterations. The results of the synthesized decoders are summarized in Table I. With the same decoding throughput, both decoders have almost the same equivalent gate counts for their logic circuits. However, the memory consumption of the 2-kB decoder nearly doubles compared with that of the 1-kB decoder. As pointed out in [10], the LDPC code decoder silicon area tends to be dominated by memory. Using a memory compiler at a 65-nm technology node, we estimate that the memory occupies about 0.74 and 1.41 mm^2 for 1- and 2-kB decoders, respectively. In comparison, the logic circuits in the decoders occupy about 0.31 mm^2.

Compared with current practice (i.e., both upper and lower pages use rate-8/9 LDPC code), the proposed scheme will result in decoder power consumption overhead. For lower page read, due to the use of a higher code rate in the lower page (e.g., code rate of 10/11 in this study) than the conventional practice (i.e., code rate of 8/9 in this study), LDPC code decoding will take more decoding iterations, leading to higher decoding power consumption. In addition, as the P/E cycle number increases so that conventional practice fails to work, the concatenated code decoding will start to be invoked and will be more frequently with further P/E cycling. This will lead to even higher power consumption. Since the power consumption is strongly dependent on P/E cycling, we carried out further simulations and power estimations to obtain the average power consumption at different P/E cycling. Table II summarizes the power consumption of the conventional design practice and the proposed strategy.

D. Impact on System Read Respond Time

When using this proposed design strategy, once the rate-r_{norm} code decoding fails for upper page data access, we must read the coding redundancy for the rate-r_{con} code stored in lower pages, leading to extra memory read latency for upper pages. Let $\tau^{(u)}_{\text{norm}}$ and $\tau^{(u)}_{\text{prop}}$ denote the upper page read latency of conventional practice and the proposed design strategy, we have

$$\tau^{(u)}_{\text{norm}} = \tau^{(u)}_{\text{sensing}} + \tau^{(u)}_{\text{trans}} + \tau_{\text{dec}}$$

where $\tau^{(u)}_{\text{sensing}}$ is the upper page sensing time, $\tau^{(u)}_{\text{trans}}$ is the Flash-to-controller data transfer time for upper page data, and τ_{dec} is the rate-r_{norm} code decoding time, i.e.,

$$\tau^{(u)}_{\text{prop}} = \tau^{(u)}_{\text{norm}} + P^{(u)}_{\text{dec\_fail}} \cdot \left(\tau^{(l)}_{\text{sensing}} + \tau^{(l)}_{\text{trans}} + n_{\text{iter-num}} \cdot \tau_{\text{dec-con}}\right)$$

where $P^{(u)}_{\text{dec\_fail}}$ is the upper page rate-r_{norm} code decoding failure probability, $\tau^{(l)}_{\text{sensing}}$ is the lower page sensing time, $\tau^{(l)}_{\text{trans}}$ is the Flash-to-controller data transfer time for the redundancy data of the rate-r_{con} code stored in lower page, $n_{\text{iter-num}}$ is the average number of concatenated code decoding iterations, and $\tau_{\text{dec-con}}$ is the latency of one concatenated code decoding iteration (i.e., the summation of rate-r_{norm} code decoding latency and the rate-r_{con} code decoding latency). Accordingly, based on the above measurement and simulation results, we can estimate the extra read latency overhead as $\eta = (\tau^{(u)}_{\text{prop}} - \tau^{(u)}_{\text{norm}})/\tau^{(u)}_{\text{norm}}$ when accessing one individual upper page under different P/E cycles, as shown in Table III for both 1- and 2-kB cases.

TABLE III
UPPER PAGE READ LATENCY OVERHEAD

P/E cycles		500	1000	1500	1800	2000
$P^{(u)}_{dec\_fail}$	1kB	10^{-5}	8×10^{-5}	0.02	0.26	0.5
	2kB	2×10^{-6}	3×10^{-5}	0.005	0.14	0.4
$N_{iter-num}$	1kB	1.00	1.00	1.07	1.16	1.29
	2kB	1.00	1.00	1.01	1.07	1.36
η	1kB	0.0	0.0	0.0	0.11	0.23
	2kB	0.0	0.0	0.0	0.06	0.19

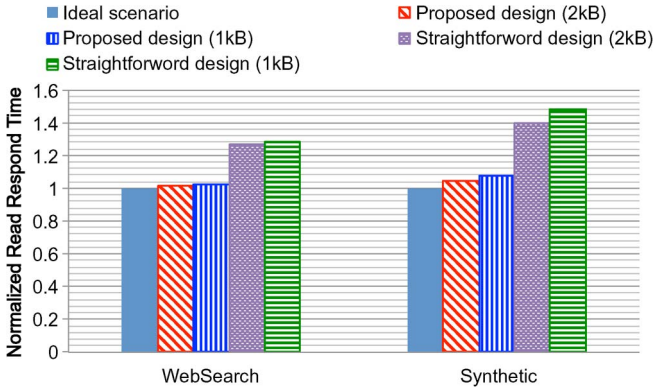

Fig. 5. Simulated SSD read respond time normalized with the ideal scenario under the P/E cycling of 1800.

The upper page read latency overhead will translate into the storage system read response time degradation in practical workload. We carried out further simulations to quantitatively evaluate the impact on storage system read response time. In particular, we carried out trace-driven simulations using the SSD module [8] in DiskSim [9] to evaluate the impact on system read response time. We use two different workload traces, including the popular WebSearch trace and a Synthetic workload in which over 90% of requests are read requests [8]. Since the probability of invoking concatenated code decoding varies with the P/E cycling, we carried out the trace-driven simulations under the P/E cycling of 1800, which corresponds to the decoding failure rate of 4% and 1% for 1- and 2-kB cases, respectively. We set the hard-decision memory sensing latency as 41 and 55 μs for lower and upper pages, respectively. Fig. 5 shows the simulation results. To facilitate the comparison, the simulated read response time is normalized to the ideal scenario when the concatenated code decoding is never invoked (i.e., the rate-$r_{\rm norm}$ code decoding never fails). For the purpose of comparison, we also show the normalized read response time when unequal error correction is realized in the straightforward manner, i.e., we directly use a rate-7/8 LDPC code and a rate-10/11 LDPC code to protect user data being stored in upper and lower pages, respectively, and we have to fetch both lower and upper pages whenever we need to read user data stored in an upper page. The results show that, when using the proposed design strategy, the read response time overhead is very small (i.e., less than 7%) for both traces. Nevertheless, when using the straightforward realization of unequal error correction, the system read response time significantly degrades, i.e., 27% for the WebSearch trace and up to 48% for the Synthetic trace.

Based on the aforementioned results, we conclude that, compared with the 1-kB LDPC code, using the 2-kB LDPC code in the proposed solution can improve the achievable P/E cycling endurance by 10% and reduce the SSD read response time overhead about by 7% (for the synthetic trace) at the cost of 64% larger silicon area. Compared with conventional design practice, this proposed solution can improve the P/E cycling endurance by 20% and incurs less than 26.9% power consumption overhead and less than 7% system read response time overhead.

V. CONCLUSION

This brief presents a simple yet effective design strategy that can enable solid-state data storage uses unequal error correction to improve P/E cycling endurance at minimal read latency overhead. The key is to employ a concatenated coding that appropriately sets the coding rates of component codes so that the probability of extra page read in unequal error correction can be minimized. In addition, by forcing all the component codes to have the same codeword length, this design strategy ensures a minimal impact on ECC decoder implementation cost. We carried out evaluations from both the coding and storage system perspectives, and the results show that this design strategy can improve the P/E cycling endurance by 20% and only incur less than 7% increase of storage system read response time at the end of Flash memory lifetime with the P/E cycling of around 1800.

REFERENCES

[1] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit error rate in NAND Flash memories," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2008, pp. 9–19.
[2] H. Zhong, W. Xu, N. Xie, and T. Zhang, "Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording," *IEEE Trans. Magn.*, vol. 43, no. 12, pp. 4117–4122, Dec. 2007.
[3] J. Yang, "Novel ECC architecture enhances storage system reliability," in *Proc. Flash Memory Summit*, Aug. 2012.
[4] X. Hu, "LDPC codes for Flash channel," in *Proc. Flash Memory Summit*, Aug. 2012.
[5] E. Yeo, "An LDPC-enabled Flash controller in 40 nm CMOS," in *Proc. Flash Memory Summit*, Aug. 2012.
[6] R. Motwani and C. Ong, "Robust decoder architecture for multi-level Flash memory storage channels," in *Proc. IICNC*, Feb. 2012, pp. 492–496.
[7] S. Tanakamaru, Y. Yanagihara, and K. Takeuchi, "Over-10x-extended-lifetime 76%-reduced-error solid-state drives SSDs with error-prediction LDPC architecture and error-recovery scheme," in *Proc. IEEE ISSCC*, Feb. 2012, pp. 424–426.
[8] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2008, pp. 57–70.
[9] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger, "The DiskSim simulation environment version 4.0 reference manual," Carnegie Mellon University Parallel Data Lab, Pittsburgh, PA, USA, Tech. Rep. CMU-PDL-08-101, May 2008.
[10] Z. Wang, Z. Cui, and J. Sha, "VLSI design for low-density parity-check code decoding," *IEEE Circuits Syst. Mag.*, vol. 11, no. 1, pp. 52–69, 2011.